Electronic Version
Stylesheet Version v1.1.1

Description

Single-Chip USB Controller Reading Power-On Boot Code from Integrated Flash Memory for User Storage

Cross Reference to Related Applications

[0001] This application is related to the co-pending application for "USB Smart Switch with Packet Re-Ordering for Interleaving among Multiple Flash-Memory Endpoints Aggregated as a Single Virtual USB Endpoint", U.S. Ser. No. 10/707,276, filed 12/2/03.

Background of Invention

[0002] This invention relates to flash memory storage, and more particularly to a serial-bus controller integrated with a parallel flash memory.

[0003] Flash memory has gained wide acceptance for its non-volatile storage, which is ideal for portable devices that may lose power, since the data is not lost when stored in the flash memory. Flash memories are constructed from

electrically-erasable programmable read-only memory (EEPROM) cells.

[0004] Rather than use a randomly-addressable scheme such as is common with dynamic-random-access memory (DRAM), many flash memories use a block-based addressing where a command and an address are sent over the data bus and then a block of data is read or written. Since the data bus is also use to send commands and addresses, fewer pins are needed on the flash-memory chip, reducing cost. Thus flash memory is often used as a mass-storage device rather than a randomly-addressable device.

[0005] Universal-Serial-Bus (USB) has become a popular standard interface for connecting peripherals to a host such as a personal computer (PC). USB-based flash-memory storage devices or "drives" have been developed to transport data from one host to another, replacing floppy disks. While large external flash drives may be used, smaller USB flash drives known as key-chain or key drives have been a rapidly growing market.

[0006] A USB flash-memory device can be constructed from a microcontroller, a flash-memory controller or interface, and one or more flash-memory chips. A serial interface on

the microcontroller connects to the USB bus to the host, and data from the serial interface is transferred through the microcontroller to the flash controller and the written to the flash-memory chips.

[0007] The microcontroller usually contains an internal ROM with a control program that is read by the internal central processing unit (CPU) of the microcontroller when the microcontroller is booted or powered up. Once initialized with the control program, the CPU can control data transfers between the serial interface and the flash controller.

[0008] Sometimes the user may desire to connect to more than one USB flash-memory device. The user can install a USB hub, and then plug the USB flash-memory devices into the USB hub's downstream ports. USB hubs allow one USB port on a host to fan out to multiple end USB devices or endpoints. A basic USB hub has a repeater that repeats data from the host to all down-stream devices, while more intelligent hubs based on the USB 2.0 standard can buffer data to different down-stream ports.

[0009] Figure 1 shows a prior-art USB hub that connects to multiple flash-memory USB endpoint devices. Host 10 includes USB host controller 12 that generates transactions to USB devices over USB bus 18 using the USB protocol.

USB hub 20 is connected to a cable containing USB bus 18. USB hub 20 fans out USB bus 18 to several downstream USB devices that connect over additional USB bus segments.

[0010] Three USB flash-memory systems 14, 15, 16 are connected to USB hub 20 by USB bus segments. USB flash-memory system 14 can be accessed by USB host controller 12 through USB hub 20. Since USB hub 20 passes all host transfers through to downstream devices, USB flash-memory system 15 is visible to host 10 as a second flash drive, while USB flash-memory system 16 is visible to the host as a third flash drive.

[0011] Users may be able to purchase a single USB flash-memory drive with a larger storage capacity and directly connect it to USB bus 18, without the need for USB hub 20. Some board manufacturers may integrate USB hub 20 together with USB flash-memory systems 14, 15, 16 on a single USB flash card or box. However, this can be expensive when USB flash-memory systems 14, 15, 16 are flash-memory chips, since each chip may have many pins. For example, a flash-memory chip with an 8-bit or 16-bit data bus may have 48 total pins. This can increase the size of the USB flash device. Power consumption is higher

due to the large number of data lines in the parallel buses to each flash-memory chip.

[0012] The transfer bandwidth of each flash-memory chip is also somewhat limited. For example, a 50 nanosecond access time for an 8-bit flash chip has a bandwidth of 160 Mbps. While this is sufficient for the USB 1.1 standard that supports a data rate of 12 Mbps, the newer USB 2.0 standard supports a data rate of 480 Mbps. Thus a wider data bus, requiring more flash-memory chips, is needed to support the faster USB 2.0 speeds. However, the additional flash-memory chips and wider buses increases board size, manufacturing cost, and product size.

[0013] What is desired is to integrate a microcontroller with a flash-memory array. It is desired to have a wide internal bus from the microcontroller to the flash-memory array to improve the data bandwidth while having few external pins to reduce cost and required board space.

[0014] It is further desired to eliminate the internal ROM on the microcontroller. Instead of booting from the internal ROM, it is desired to use a control program stored in the flash-memory array. However, it is also desired to use a block-addressed rather than a randomly-addressable array for the flash storage.

Brief Description of Drawings

[0015] Figure 1 shows a prior-art USB hub that connects to multiple flash-memory USB endpoint devices.

[0016] Figure 2 is a block diagram of a flash microcontroller integrated with USB flash mass storage blocks.

[0017] Figure 3 shows external pin connections to a USB single-chip flash device.

[0018] Figure 4 is a block diagram of the USB flash microcontroller inside the USB single-chip flash device.

[0019] Figure 5 is a block diagram of a flash mass storage block.

[0020] Figure 6 is a diagram of the first page in the flash mass storage block.

[0021] Figure 7 is a flowchart of a power-up sequence that does not require a ROM on the internal microcontroller bus.

[0022] Figure 8 is a flowchart of the microcontroller executing the boot loader program to load the control program.

[0023] Figure 9 is a flowchart of upgrading the control program.

[0024] Figure 10 shows the use of a USB single-chip flash device with a host.

[0025] Figure 11 shows a larger-capacity flash drive with multiple USB single-chip flash devices.

[0026] Figure 12 shows the single-endpoint USB switch in more

detail.

DETAILED DESCRIPTION

[0027] The present invention relates to an improvement in flash memory. The following description is presented to enable one of ordinary skill in the art to make and use the invention as provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed.

[0028] Figure 2 is a block diagram of a flash microcontroller integrated with USB flash mass storage blocks. USB host controller 12 for host 10 sends transactions over USB bus 18 to USB single-chip flash device 40. USB flash microcontroller 30 on USB single-chip flash device 40 receives and responds to transaction from host 10 over USB bus 18. USB flash microcontroller 30 has a serial interface that acts as the final USB endpoint for transactions on USB bus 18 from host 10.

[0029] USB flash microcontroller 30 also contains a flash-memory controller that sends data on internal flash buses 28 to flash mass storage blocks 22, 23, 24. Flash mass storage blocks 22, 23, 24 respond to internal requests from USB flash microcontroller 30 by transferring data over internal flash buses 28.

[0030] Since internal flash buses 28 are internal to USB single-chip flash device 40, external pins are not required for the interface to flash memory. A wider internal bus of 32, 64, or 128 bits can be used for internal flash buses 28, improving data bandwidth. However, flash mass storage blocks 22, 23, 24 are not randomly accessible. Instead, a command and an address are transferred as data over internal flash buses 28 to indicate a block of data to transfer from flash mass storage blocks 22, 23, 24. Thus flash mass storage blocks 22, 23, 24 are block-addressable mass storage rather than random-access memory (RAM).

[0031] Flash mass storage blocks 22, 23, 24 may be aggregated together by USB flash microcontroller 30, which maps and directs data transactions to selected flash storage blocks 22, 23, 24. Since USB flash microcontroller 30 performs memory management, flash storage blocks 22, 23, 24 appear as a single, contiguous memory to host 10.

[0032] Figure 3 shows external pin connections to a USB single-chip flash device. USB single-chip flash device 40 connects to the host through USB bus 18, which has power (Vcc) and ground and differential data lines D+, D-. Data is sent over differential data lines as analog NRZI data.

[0033] A write-protect (WP) pin connects externally to switch 25. Switch 25 can be switched by the user to indicate write-protect or write-enable modes of the flash memory inside USB single-chip flash device 40.

[0034] A status output pin for a light-emitting diode (LED) can be included on some embodiments. The status-LED pin can drive LED 41 to indicate a status of USB single-chip flash device 40. For example, LED 41 can be made to blink when the internal flash memory is being written so that the user does not unplug the device before writing is completed.

[0035] Relatively few pins are needed for USB single-chip flash device 40. Since USB bus 18 is only 4 pins, including power and ground, as little as 5 pins are needed when no LED signaling is required. All commands, addresses, status, and data are carried as serial analog data over the differential data lines in USB bus 18. Additional power and ground pins, or pins for other functions could be added,

but packages with fewer than 10 pins are relatively inexpensive and require little board space. Power consumption is also reduced, since fewer higher-capacitance external signals are driven by USB single-chip flash device 40.

[0036] Figure 4 is a block diagram of the USB flash microcontroller inside the USB single-chip flash device. Internal bus 66 connects CPU 52 with RAM 56, serial-data buffer 64, direct-memory access (DMA) engine 58, and flash-memory controller 60. CPU 52 executes instructions from RAM 56, while DMA engine 58 can be programmed to transfer data between serial-data buffer 64 and flash-memory controller 60. CPU 52 can operate on or modify the data by reading the data over bus 66. RAM 56 can store instructions for execution by the CPU and data operated on by the CPU.

[0037] Serial transceiver 54 connects to the differential data lines D+, D- of USB bus 18 and contains both a differential receiver and a differential transmitter. Data is encoded or decoded using NRZI encoding. Bit stuffing can be used to align data. An interrupt to CPU 52 can be generated when a start-of-packet sequence is detected on USB bus 18. CPU 52 can then execute a routine to handle the interrupt and process the new packet.

[0038] Serial engine 50 can perform higher-level functions such as checking cyclical-redundancy-check (CRC) checksums, locating packet identifiers, end-of-packet markers, higher-level frame markers, and converting serial data to parallel data words. The transmit and receive data is stored in serial-data buffer 64. Commands and addresses from the USB packets can also be stored in serial-data buffer 64, but is read by CPU 52 to determine what operation to perform rather than sent directly to flash-memory controller 60.

[0039] Flash-memory controller 60 includes flash data buffer 68, which contains the commands, addresses, and data sent over internal flash buses 28 to flash mass storage blocks 22, 23, 24. Data can be arranged in flash data buffer 68 to match the bus width of internal flash buses 28, such as in 32 or 64-bit words. DMA engine 58 can be programmed by CPU 52 to transfer a block of data between flash data buffer 68 and serial-data buffer 64.

[0040] Error-corrector 62 can read parity or error-correction code (ECC) from flash mass storage blocks 22, 23, 24 and perform data corrections. The parity or ECC bits for data in flash data buffer 68 that is being written to flash mass storage blocks 22, 23, 24 can be generated by error-

corrector 62.

[0041] Flash programming engine 67 can be a state machine that is activated on power-up reset. Flash programming engine 67 programs DMA engine 58 within the address of the boot loader code in the first page of flash mass storage block 22, and the first address in RAM 56. Then flash programming engine 67 commands DMA engine 58 to transfer the boot loader from flash mass storage block 22 to RAM 56. CPU 52 is then brought out of reset, executing the boot loader program starting from the first address in RAM 56. The boot loader program can contain instructions to move a larger control program from flash mass storage block 22 to RAM 56. Thus USB flash microcontroller 30 is booted without an internal ROM on internal bus 66.

[0042] Figure 5 is a block diagram of a flash mass storage block. Flash mass storage block 22 is on the same chip as USB flash microcontroller 30 and is connected by internal flash bus 28 and a control and status bus. Flash memory array 80 contains the flash or EEPROM memory cells arranged as data words that match the width of internal flash bus 28, such as 32, 64, or 128 bits wide. Address signals for the rows and columns are generated by address buffers

and decoder 82, which is loaded with an address that is sent over internal flash bus 28 and through global buffer 78.

[0043] A command sent over internal flash bus 28 is latched into command register 84 from global buffer 78. This command is decoded and activates control logic 86 to carry out a sequence of operations and generate a sequence of internal control to perform the operation, such as reading or erasing memory cells or latching an address or reporting a status back. Flash mass storage block 22 may be powered down when not in use by a command that activates power manager 88.

[0044] Data written into flash mass storage block 22 arrives on internal flash bus 28 and is initially latched by global buffer 78. The data is then sent to data buffers 78 to be driven onto flash memory array 80 for storage. Data may be temporarily held in data/cache registers 74 while being written to flash memory array 80.

[0045] Commands indicate a block of data to be transferred rather than individual bytes or words. For example, a page of 512 to 528 bytes or a multiple thereof could be transferred by a read or write command. Since one block address is transferred over internal flash bus 28 for all bytes

in the block, the overhead time for the address is spread over many bytes of data and thus does not significantly reduce performance. Also, when blocks are aligned, the block address can have fewer address bits than a full byte or word address.

[0046] Some flash memory chips have a "first-page auto-read" mode that allows the first page of memory to be read after power up, before a command or an address is sent to the flash-memory chip. A similar mode can be provided for flash mass storage block 22 to allows the first page of flash memory to be read without sending an initial address or command. Power manager 88 detects the power-on condition and sets the address in address buffers and decoder 82 to the first address of the first page in flash memory array 80. When a read signal is sent over the control/status bus, a read command is executed with successive data words from this first page being read from flash memory array 80 to internal flash bus 28.

[0047] Figure 6 is a diagram of the first page in the flash mass storage block. This page is automatically read on power-up starting with the first data bytes. Flash memory space 200 has stored in it a first copy of boot loader program 202, starting at the first address on the first page. Backup

copy 204 of the boot loader program follows boot loader program 202 and can be loaded in case of error.

[0048] Two copies 206, 208 of a control program are stored in flash memory space 200. When one copy contains an error, the other copy can be loaded. The remaining addresses on the first page, and on subsequent pages of flash memory space 200 are used for user storage 210. Data from the host can be stored in user storage 210.

[0049] Microcontroller ROM Eliminated

[0050] Since the boot loader program and the control program reside on the same flash mass storage blocks as the user or host data, a separate memory is not required for the boot loader or for the control program. Microcontrollers typically contain a ROM that contains the control program or boot loader. The ROM is read by the CPU over the internal CPU bus at power-up. The inventors have realized that this ROM is not needed for the USB single-chip flash device.

[0051] Even though flash mass storage block 22 is not randomly-addressable, a first-page auto-read feature of flash memory can be used to locate the boot code on power-up. The boot code then can be transferred to the microcontroller's RAM for execution by the microcontroller CPU.

[0052] Figure 7 is a flowchart of a power-up sequence that does not require a ROM on the internal microcontroller bus. Initial load sequence 100 is performed by a hardware state machine, flash programming engine 67 (Fig. 4), and then by DMA engine 58. A boot-loader program is read from the flash mass storage block through the flash-memory controller and written to the RAM on the internal bus before the microcontroller's CPU is taken out of reset.

[0053] At power-on the CPU in the USB flash microcontroller remains in the reset state, step 102. The power manager in flash mass storage block 22 loads the starting address of the first page of flash memory into the address buffers and decoders, step 104, to enable auto-reading of the first page of flash memory. A command and an address do not have to be transferred to the flash memory for this initial auto-read of the first page. While the flash memory is not randomly-addressable, it is read sequentially starting from the first byte or word of the first page.

[0054] Flash programming engine 67 in flash-memory controller 60 comes out of reset and programs the DMA engine with source and target device addresses and an initial transfer length. The DMA engine begins transferring the boot code, step 106. The DMA engine reads from the first page

of the flash memory, starting with the first data word, step 108. Data is read from flash mass storage block 22 through flash-memory controller 60, across internal bus 66 to RAM 56 (Fig. 4).

[0055] Data words continue to be transferred by the DMA engine until the initial transfer length is reached, step 110. The last bytes of the boot loader program can contain a checksum that is compared to a checksum generated by the DMA engine, step 112. When the checksums do not match, an error handler routine hardwired into flash programming engine 67 is activated, which can then read the backup copy of the boot loader program. When the checksum is good, the boot loader program has been successfully transferred from the first page of flash mass storage block 22 to the microcontroller's RAM. The microcontroller's CPU can then be brought out of reset. Flash programming engine 67 or the DMA engine can keep track of which copy of the boot loader is the active copy, or indicate to the control program when an error occurred. For example, flash programming engine 67 could contain a status register that the CPU can later read after booting.

[0056] Figure 8 is a flowchart of the microcontroller executing the boot loader program to load the control program. The

boot loader's instructions are executed by the CPU during control-program load sequence 120.

[0057] Once the boot loader program has been transferred to the RAM, the CPU in the microcontroller is taken out of reset. The CPU reads the first instruction at the first address in the RAM and begins executing the boot loader's instructions in the RAM, step 122. The boot loader contains CPU instructions to read the control program from the first page of flash mass storage block 22. This control program is read and transferred to the microcontroller's RAM, step 124.

[0058] A checksum at the end of the control program is checked against an expected value such as zero, step 126. If the checksum does not match the expected value, the back-up copy of the control program is loaded from the flash mass storage block, step 128. Once the checksum is correct after loading either the first or backup copy of the control program, the memory space in the RAM used by the boot loader program can be released and over-written, step 130. The CPU then begins executing instructions from the control program. Normal operation can begin, such by as initializing the serial engine.

[0059] Control Program Upgrading - Fig. 9

[0060] Figure 9 is a flowchart of upgrading the control program. One benefit of storing the control program in flash mass storage block 22 rather than in a ROM on the microcontroller is that the control program can be upgraded by writing to the first page of the flash memory. Hardwired ROM cannot be upgraded: discovery of a program bug may require re-tooling the chip photo masks.

[0061] Upgrade routine 140 can be activated when a control-program-upgrade request packet is received from the host. The CPU reads both copies of the control program in flash memory to determine which copy is the active copy. The active copy in RAM can be compared to the two copies in flash memory to determine which copy matches the active copy in RAM. The non-matching copy is the last-known-good copy and is the copy replaced by the upgraded control program if the active copy is determined to be as reliable as the last known good copy. Otherwise, the active copy can be over-written.

[0062] One of the two copies of the control program is indicates as the active copy and the other as the backup copy. For example, a flag stored in memory could designate which copy is the active copy. Alternately, the flag can be eliminated by having the executing control program read both

copies from the flash memory and compared each to the copy in the RAM that the CPU is executing from to determine which copy is the active copy. Version numbers could also be read and compared. One of the two copies of the control program in the flash mass storage block is selected for replacement, step 142. The upgraded copy of the control program sent by the host then over-writes the selected copy in the flash mass storage blocks, step 144. The upgraded program written in can be verified if desired.

[0063] The control program is thus field loadable and upgrade-able though the serial bus to the host. New features can be added to the control program, and bugs can be fixed. The control program code is no longer frozen at the time of manufacture. This is a significant advantage over a microcontroller with a hard-wired ROM. The boot loader program could also be upgraded, but this is not likely to be needed due to the relative simplicity of the boot loader program. Eliminating the ROM also can reduce manufacturing cost.

[0064] Figure 10 shows the use of a USB single-chip flash device with a host. Host 10 conducts USB transactions with USB single-chip flash device 40 by sending USB packets from

USB host controller 12 over USB bus 18. Data from the host is stored in the flash mass storage blocks inside USB single-chip flash device 40. The control program is also read from the first page of the flash mass storage blocks during boot up and transferred to the USB flash microcontroller's internal RAM so that the CPU can execute the RAM's copy of the control program.

[0065] While the user can plug a single USB single-chip flash device into USB bus 18, sometimes a higher flash-memory storage capacity is desired, such as for transferring larger amounts of data. The user could install a standard USB hub on USB bus 18 and then plug multiple USB single-chip flash devices 40 into the hub. However, a single device with multiple USB single-chip flash devices 40 may be more desirable.

[0066] Figure 11 shows a larger-capacity flash drive with multiple USB single-chip flash devices. Multiple USB single-chip flash devices 40 are connected to USB switch 70. A manufacturer could construct the multi-chip device by placing USB switch 70 and the multiple USB single-chip flash devices 40 on a single printed-circuit board (PCB) substrate. The small size and low pin count of USB single-chip flash device 40 makes it ideal for such an application, allowing

larger-capacity flash-storage devices to be economically built.

[0067] While a standard USB hub could be substituted for USB switch 70, using a smart switch rather than a hub has several advantages. USB switch 70 can operate in a single-endpoint mode where host 10 sees just one USB endpoint device, USB switch 70, which appears as a single USB flash device having a larger memory size than each individual USB single-chip flash device 40. Thus USB switch 70 aggregates downstream flash memory in USB single-chip flash devices 40 and presents one virtual USB flash drive to the host.

[0068] The USB single-chip flash device 40 is ideally suited for expansion and scaling to larger flash-memory capacities. The low pin count and serial interface to USB single-chip flash device 40 reduces board space and enables manufacture of small-physical-size flash devices with larger storage capacities. Since USB single-chip flash devices 40 are electrically and functionally compatible with the widely-used USB standard, they are easily combined with various USB hubs and other USB devices.

[0069] Figure 12 shows the single-endpoint USB switch in more detail. More details of USB switch 70 are provided in the

co-pending application for "USB Smart Switch with Packet Re-Ordering for Interleaving among Multiple Flash-Memory Endpoints Aggregated as a Single Virtual USB Endpoint", filed 12/2/03, U.S. Ser. No. 10/707,276.

[0070] USB switch 70 connects to host USB bus 18 through USB upstream interface 34. USB switch 70 connects to down-stream USB flash storage blocks over secondary USB buses 27 through USB downstream interfaces 46, 47, 48. USB interfaces provide physical signaling, such as driving and receiving differential signals on differential data lines of USB buses, detecting or generating packet start or stop patterns, checking or generating checksums, and higher-level functions such as inserting or extracting USB device addresses and packet types and commands.

[0071] Mode logic 26 senses the voltage on mode pin 29, which can be pulled down to ground externally for single-endpoint mode, or pulled high with a pull-up resistor for standard USB hub mode. Mode logic 26 activates USB switch 70 to operate as a hub or as an aggregating and virtualizing switch. For hub mode, data is buffered across virtual USB bridge 32 from the host to one of virtual USB bridges 42, 43, 44 to flash memory. Internal bus 38 al-lows data to flow among virtual USB bridge 32 and USB

bridges 42, 43, 44. The host and the endpoint may operate at the same speed (USB low speed (LS), full speed (FS), or high-speed (HS)), or at different speeds. Buffers in internal virtual USB bridge 32 can store the data.

[0072] Virtual storage processor 140 provides re-mapping and translation services to transaction manager 36. For example, logical addresses from the host can be looked up and translated to physical device addresses in USB flash storage blocks 22, 23, 24. Thus the details of physical flash memory in flash mass storage blocks 22, 23, 24 are hidden from the host.

[0073] When operating in single-endpoint mode, transaction manager 36 not only buffers data using virtual USB bridge 32, but may also re-order packets for transactions from the host. A transaction may have several packets, such as an initial token packet to start a memory read, a data packet from the memory device back to the host, and a handshake packet to end the transaction. Rather than have all packets for a first transaction complete before the next transaction begins, packets for the next transaction can be re-ordered by USB switch 70 and sent to the memory devices before completion of the first transaction. This allows more time for memory access to occur for the next

transaction. Transactions are thus overlapped by re-ordering packets.

[0074] Packets sent over secondary USB buses 27 can be re-ordered relative to the packet order on host USB bus 18. Transaction manager 36 may overlap and interleave transactions to different USB flash storage blocks, allowing for improved data throughput. For example, packets for several incoming USB transactions from the host are stored in virtual USB bridge 32 or an associated buffer (not shown). Transaction manager 36 examines these buffered transactions and packets and re-orders the packets before sending them over internal bus 38 to a downstream USB flash storage block.

[0075] A packet to begin a memory read of a flash block through USB bridge 43 may be re-ordered ahead of a packet ending a read of another flash block through USB bridge 42 to allow access to begin earlier for the second flash block.

[0076] ALTERNATE EMBODIMENTS

[0077] Several other embodiments are contemplated by the inventors. For example different numbers and arrangements of USB flash storage blocks can connect to the USB switch. Rather than use USB buses, other serial buses may be used such as PCI Express, ExpressCard, Firewire (IEEE

1394), serial ATA, serial attached small-computer system interface (SCSI), etc. For example, when PCI Express is used, additional pins for the PCI Express interface can be added or substituted for the USB differential data pins. PCI express pins include a transmit differential pair PET+, PET-, and a receive differential pair PER+, PER- of data pins. A multi-bus-protocol chip could have an additional personality pin to select which serial-bus interface to use, or could have programmable registers. ExpressCard has both the USB and the PCI Express bus, so either or both buses could be present on an ExpressCard device.

[0078] The mode logic could sense the state of a pin only at power-on rather than sense the state of a dedicated pin. A certain combination or sequence of states of pins could be used to initiate a mode change, or an internal register such as a configuration register could set the mode.

[0079] The microcontroller and USB switch components such as the serial engine, DMA, flash-memory controller, transaction manager, and other controllers and functions can be implemented in a variety of ways. Functions can be programmed and executed by the CPU or other processor, or can be implemented in dedicated hardware, firmware, or in some combination. Many partitioning of the functions

can be substituted.

[0080] The host can transfer standard USB packets to the serial engine during a transaction, such as a token, data, and handshake packet. Other packet types or variations of these types can be defined for special purposes. These packets may include a flash-controller-request packet, a flash-controller-reply packet, a boot-loader-request packet, a boot-loader-reply packet, a control-program-request packet, a control-program-reply packet, a flash-memory-request packet, and a flash-memory-reply packet. The flash-memory request/reply packets may further include the following request/reply packet pairs: flash ID, read, write, erase, copy-back, reset, page-write, cache-write and read-status.

[0081] The boot loader and control program could be larger than the first page of the flash memory or could be less than the size of the first page. The entire first page or just a portion of the first page could be transferred. Different page sizes could be substituted. One of the flash mass storage blocks 22, 23, 24 could be designated to respond during the first-page auto-read sequence and not the others. Backup copies could be kept in the others of flash mass storage blocks 22, 23, 24. More than one backup

copy could be kept. The control program could call still other programs or routines. More complex memory management could be added.

[0082] The CPU could be taken out of reset but kept in a hold state or otherwise suspended from execution while the boot loader and control program are being transferred from the flash mass storage block. Several re-tries of reading or writing the boot loader or control program could be attempted.

[0083] Wider or narrower data buses and flash-memory blocks could be substituted, such as 8, 16, 32, 64, 128, 256-bit, or some other width data channels. Alternate bus architectures with nested or segmented buses could be used internal or external to the microcontroller. Two or more internal buses can be used in the USB flash microcontroller or in the USB switch to increase throughput. More complex switch fabrics can be substituted for the internal buses.

[0084] The flash mass storage blocks can be constructed from any flash technology including NAND, NOR, AND, or multi-level-logic memory cells. Data striping could be used with the flash mass storage blocks 22, 23, 24 in a variety of ways, as can parity and error-correction code

(ECC). Packet re-ordering can be adjusted depending on the data arrangement used to prevent re-ordering for overlapping memory locations. The USB switch can be integrated with other components or can be a stand-alone chip. The USB switch could also be integrated with USB single-chip flash devices 40.

[0085] The abstract of the disclosure is provided to comply with the rules requiring an abstract, which will allow a searcher to quickly ascertain the subject matter of the technical disclosure of any patent issued from this disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims. 37 C.F.R. Sect. 1.72(b). Any advantages and benefits described may not apply to all embodiments of the invention. When the word "means" is recited in a claim element, Applicant intends for the claim element to fall under 35 USC Sect. 112, paragraph 6. Often a label of one or more words precedes the word "means". The word or words preceding the word "means" is a label intended to ease referencing of claims elements and is not intended to convey a structural limitation. Such means-plus-function claims are intended to cover not only the structures described herein for performing the function and their

structural equivalents, but also equivalent structures. For example, although a nail and a screw have different structures, they are equivalent structures since they both perform the function of fastening. Claims that do not use the word "means" are not intended to fall under 35 USC Sect. 112, paragraph 6. Signals are typically electronic signals, but may be optical signals such as can be carried over a fiber optic line.

[0086] The foregoing description of the embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.